Self-Organizing Networks in LTE: a Q-learning approach to ABS optimization

Andrea Maracani, Marco Rossanese, Davide Talon
Department of Information Engineering, University of Padova – Via Gradenigo, 6/b, 35131 Padova, Italy
Email: {andrea.maracani,marco.rossanese,davide.talon}@studenti.unipd.it

***Abstract*—In this paper we introduce a possible machine-learning approach for the enhanced inter-cell interference coordination (eICIC) in a heterogeneous network (HetNet), where macro and micro cells optimize their downlink transmission in a self-adaptive manner. The idea is to exploit the Q-learning algorithm to guarantee a fair resource sharing between micro and macro cells and so to make the system works in the best performance possible. Indeed this specific reinforcement learning technique is used to find out the optimal policy that leads to the right ABS pattern setting, designed specifically for each scenario. Hence evaluating only a few parameters and changing dynamically the ABS pattern we aim to cope the possible alterations on a realistic environment. Finally, three different ABS policies are evaluated over many traffic models in term of the average throughput per user.**

***Index Terms*—SON, Self Optimizing Networks, LTE, eICIC, ABS, reinforcement learning, Q-learning, MONSTeR**

I. Introduction

The great spreading of mobile devices all over the world represents the fastest adoption of any technology that our society has ever experienced, faster than the Internet and the earlier generations of mobile communications. Now tablets, android devices, iPhones, application stores, social media and the data exchanges between end-users and clouds are all growing at exponential speed. Providing the necessary bandwidth and capacity to the people so they can keep the pace of this growth is a fascinating challenge; in order to achieve that, an increased network densification is required together to a full exploitation of the simultaneous presence of micro (coverage radius around 10m-300m) and macro (coverage radius up to 20km) cells, the so called heterogeneous networks (HetNet). Figure 1 illustrates a typical HetNet configuration.
In 2016, global mobile data traffic amounted to 7 exabytes (EB)[1] per month; in 2021, mobile data traffic worldwide is expected to reach 49 exabytes per month at a compound annual growth rate of 47 percent [5]. These challenges can be solved in a cost-effective, efficient and human-handleable way only through the use of more automated and autonomous systems, such as Self-Organizing Networks (SON): the goal is to minimize the human intervention in the planning, deployment, optimization and maintenance activities of these new networks. Such a SON conceptually must own, as explained by [7], the following capabilities:

- Self-Planning: process of identifying the parameter settings of new network elements (like radio parameters of a new eNodeB or a table of neighbour nodes);
- Self-Deployment: preparation, installation, authentication and delivery of a status report of a new network node in order to get a "plug and play" approach for each new device;
- Self-Healing: execution of the routine actions that keep the network operational and/or prevent problems (this includes the necessary software and hardware upgrades);
- Self-Optimization is defined as the utilization of measurements and performance indicators collected by the User Equipments (UEs) and the base stations in order to auto-tune the network settings.

[1] 1 EB = $8 \cdot 10^{18}$ bit

Our work is focused on the last point of the list, more precisely on the improvement of the signal quality and consequently the throughput in a LTE system, trying to minimize the interference between micro and macro cells with an adaptive coordination in downlink transmissions of the antennas (mainly controlling the transmit power patterns).
The problem we are facing consists in finding the optimal trade-off between assigning the radio resources to high-power nodes (macro) or low-power nodes (micro). The latter has an higher capacity and consequently an user connected to it will have better performances; however, low-power nodes are severely affected by the interference from the high-power nodes, due to fact that they share the same frequency band with the macro. In other words, downlink micro transmissions to its UEs could be sorely degraded by high power macro transmission; besides UEs that are close to a micro could end up associating to a macro due to the higher power strength received from the stronger node.
Accounting the above scenario, the micro could be left under-utilized and this would turn out to be a bad exploitation of the resources deployed appositely to enhance the connectivity in some areas, with a following shortage of the user performance in comparison with achievable potential capacity.
Addressing a fair resource sharing, a solution could be stop macros from all the transmissions (some exception signals, like beacons, are always active), for a certain amount of time: this approach is adopted by LTE 3GPP standard and it is known as enhanced Inter Cell Interference Coordination (eICIC). The period in which macros are "silent" is called Almost Blank Subframe (ABS), period over which micros can transmit with reduced interference.
Time is slotted in frames of 10 ms and each one of these is split into 10 subframes, so a macro decides to mute itself in a certain subframe following the ABS mask, that's the pattern of 0's and 1's where 1 indicates the macro's inactivity.

The goal is to discover an adaptive ABS mask that maximizes the UEs' performance and it is found as the result of an optimization problem. Our idea is to use a reinforcement learning (RL) approach, the Q-learning technique, in order to reach the fairness between the shared resources and to set the best ABS mask in a self-tuning manner for each specific scenario.
Thanks to the LTE environment simulator MONSTeR (MObile Networks SimulaToR) [1], a framework built around the LTE system toolbox available in MATLAB, we simulate a great amount of data traffic with the purpose of training the Q-learning: after this period the method involved will yield the best policy for setting the considered mask.
Simulations prove that our modeling choice is not the optimal one but could be a good starting point to develop eICIC optimization through Q-Learning, indeed the proposed algorithm performs poorly respect the baseline solutions.
The rest of the paper is organized as follows: Section II presents how eICIC is addressed in literature. Section III introduces the adopted system model, specifying the necessary theoretical background to explain the steps that leaded us to define it. Section IV states all the simulation settings used in the environment and testing scenario; finally Section V concludes the work analyzing the achieved results and comparing the obtained policy with benchmark ones.

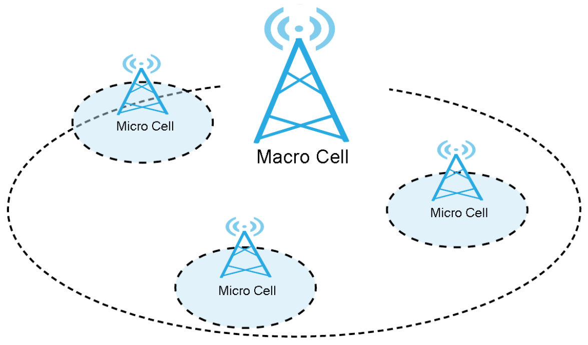

Figure 1: Example of a typical HetNet with a macro and some micros used for throughput enhancement in high density utilization area; a micro is placed on the edge of the coverage area for improving throughput of the border UEs.

II. Related Work

In this section we will overview the main proposals in the literature.
First of all we must take in account the solution exposed in [3] to protect the downlink micro transmissions. Alongside the concept of ABS period they introduced the so called Flexible User Association and Cell selection bias (CSB). As we already know, whenever a UE needs to select a suitable cell for association it chooses the one with the strongest signal and therefore the utilization of the micro most of the time is neglected. LTE has introduced the bias value α_i that is broadcasted to all the UEs. Hence the UE will choose the right cell maximizing the sum between α_i and the received signal power P_i over each i-th cell in the system; obviously higher bias values are assigned to the micros.
Deb *et al.* [3] proposed two algorithms, one for the optimal ABS mask (OPT-ABS) and one for defining the right value for the CBS. The former is split into two parts: the first to search the number of optimal subframes in which the macros are mute and second how to define the optimal ABS pattern. This algorithm turned up to be a NP-hard computational problem, however they proofed that with a clever breakdown in some defined steps it's implementable and the complexity is scalar with number of cells. Instead the latter algorithm compute the biases so that the "association error" (as compared to optimal association) is minimized. This technique shows a shortcoming since that the beforehand knowledge of the best cell is an information not always available and so it needs to be evaluated for each scenario.
Another noteworthy paper is surely [8], here the authors tried to put into practice a similar reinforcement learning approach. The state is made of two parameters, the Signal to Intererence Noise Ratio (SINR) for the player in relation with the macro and the micros, nevertheless these are not values that the eNodeB takes normally into account and we are aiming to use only the minimum number of information that are already exchanged (all the info shared between eNodeB and UE are listed in the 3GPP release 36 [4]) reducing the system overhead.

III. System Model

In this section we will explain all the operative assumptions, with the theoretical motivations, that we took into account for our proposal.

A. Problem modelization

As stated in [9], the main book about RL: the reinforcement learning problem is meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner, who is the one that makes the decisions is called the agent and in this scenario will be the macro. The thing it interacts with, comprising everything outside the agent, is called the environment, part played here by the simulator MONSTeR. These interact continually, the agent selecting actions and the environment reacting to those actions and presenting new situations to the agent. The environment also yields rewards, numerical values that evaluate the goodness of the last action taken and that the agent tries to maximize over time. Defining S as the set of all possible states and A_s as the available actions at state s, the problem is modelled as a finite Markov Decision Process (MDP) defined as a 5-tuple:

$$(S, A_s, P_a(s_t, s_{t+1}), R_a(s_t, s_{t+1}), \gamma)$$

where $P_a(s_t, s_{t+1})$ represents the probability to move from state s_t to state s_{t+1} with action $a \in A_s$, $R_a(s_t, s_{t+1})$ describes the obtained reward from the transaction s_t towards s_{t+1} with action $a_t \in A_s$ and the discount factor γ reflects how to weight future actions' reward. Therefore the environment is represented as a Markov chain in which each transaction relies on the agent's action and environment's response. We need a small as possible state space to reduce the complexity of the MDP, but at the same time each state should contains enough information to preserve the Markov's property. Hence, at each time step t, the agent receives a representation of the environment's state S_t, here defined by three parameters:

- the ratio between the number of users in the macro and in the micro, quantized in four values which indicates if there are many more users in the macro, a little bit more in the macro, a little bit more in the micro or many more in the micro;
- the ratio between the throughput of the macro cell and the mean value of the throughtput of all micro cells, quantized in five values;
- the number of ABS is quantized in six values where each of them stays for the amount of subframes in which the macro cell stays silent. In particular we just consider ABS masks composed by an even number of silent subframes, that are actually six.

Then, for the curse of dimensionality, the total number of possible states in our environment is exactly $4 \cdot 5 \cdot 6 = 120$ which allow us to handle the convergence problem. Depending in which state the agent is, it selects an action A_t from the set of all actions $A = \{-2, 0, 2\}$ that will modify the ABS mask by this amount. In the states where the ABS mask is equal to 0 there is not the decreasing-action (-2) and in the states with ABS mask equal to 10 there is not the increasing action (+2). The reward is a number between 0 and 10 and it must be an high value when the ratio between the thoughput per user in the macro cell and the throughput per user in the microcells is close to one. In fact this condition is index of fairness between the downlink transmissions for users in the macro cell and users in the micro cells. In particular the ratio we consider is:

$$r(n^m, n^M, S^m, S^M) = \left(\frac{S^M/n^M}{S^m/n^m}\right) = \left(\frac{S^M \cdot n^m}{S^m \cdot n^M}\right)$$

And so, the reward function we consider is:

$$\begin{aligned} &Reward(n^m, n^M, S^m, S^M) \\ &= \begin{cases} 10 \cdot r(n^m, n^M, S^m, S^M), & \text{if } r < 1 \\ max(10 \cdot (2 - r(n^m, n^M, S^m, S^M), 0), & \text{if } r > 1 \end{cases} \end{aligned}$$

where m stands for micro and M for macros and it shows the discrepancy between the actual system condition and the equilibrium we're aiming to.
At each time step, the agent maps from states to probabilities of taking each possible action. This mapping is called policy and is denoted as $\pi : S \times S \times A \to R$. Reinforcement learning specifies how the agent changes its policy in relation of its experience. The agent's goal is to learn actions that maximize the discounted reward, that is the total amount of reward it gains on the long term:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Actions cannot follow a greedy selection, a myopic choice could yield the MDP to achieve a sub-optimal solution, that is in a local minimum point.

B. Q-learning technique

Q-learning algorithm was proposed by Watkins in [10] and consists in maintaining a table of estimated expected long-run reward, called indeed Q-values and written as $Q(S, A)$, for each state-action couple. This technique is classified between the off-policy methods, in other words it evaluates or improves a policy different from that used to generate the data, as explained by [9]. So the idea to exploit two policies, one that is learned about and that becomes the optimal policy (target policy), and an exploratory one used to estimate the Q-values (exploration policy). Despite that the most used policies to implement the latter are the ϵ-greedy or the softmax, we decided to use a totally random selection policy.
After Q-values have been enough discovered, the Q-learning function points out how to update the Q-value at each step and it's described as follows:

$$Q(S_{t+1}, A_{t+1}) = Q(S_t, A_t) + \alpha[R(S_t, A_t) - Q(S_t, A_t)]$$

where R is the maximized expected reward and α, indicates with which rates we take actions towards the best Q-values achievable.

IV. Simulation Settings

In the present work in order to evaluate the proposed algorithm we run out an extensive MATLAB simulation using MONSTeR. We generated an urban scenario of 4000 squared meters with 4 horizontal and 3 vertical streets and among them we located buildings with height uniformly distributed from 20 to 50 meters. Then only macro eNodeB is placed at the center of the city and 4 micros are equally distributed on a circumference centered in the macro with radius of 40 m; the former has height 35 m and 50 Resource Blocks (RB) and the latter have height 25 m and 25 RBs. The mobility of the users is generated randomly, 15 users move with velocity 70 m/s[2] along sidewalks and at each cross have the same probability to turn left/right or to go straight ahead.
To simulate the channel and the wireless communication we considered the Winner II channel model [2] with urban propagation scenario implemented inside MATLAB Communications System Toolbox. To assess the worth of our algorithm we considered as main metrics the Bit Error Rate (BER) and the average throughput of involved users, therefore we selected as benchmark two ABS choice policy from the literature in three different traffic scenarios:

- **Static ABS**: the number of Almost Blank Subframes remains constant to 4 over the entire simulation and cells use the same ABS mask [0,1,0,1,0,1,0,1,0,0];
- **Random ABS**: at each frame, macro cell randomly selects the ABS mask choosing over three possible action: increase, decrease or keep constant the number of almost blank subframes. Each action has the same probability, however if we are dealing with 0 ABS the only possible actions are to keep constant or increase with probability respectively 1/3 and 2/3. The case of 10 ABS is handled

[2]This high speed is imposed by the need of computing a sufficiently number of handovers in a frame.

analogously but with decreasing action in lieu of increasing.

Table I sums up the ABS policy parameters.

Table I: ABS policy algorithm parameters.

ABS POLICY	PARAMETER	VALUE
Random policy	$Prob[a_{incr}]$	1/3
	$Prob[a_{cost}]$	1/3
	$Prob[a_{decr}]$	1/3
Static	N_{ABS}	4
Q-learning	γ	0.9
	α	0.4

We compared the system over three different traffic models. In the simple web browsing model, we are simulating a realistic human behavior during a web surfing period: packet request events from an user is modelled as a Poisson process of rate λ, which is taken from a Gaussian distribution $\lambda \sim \mathcal{N}(75, 10)$. So whenever a generation event occurs a packet file is created with a random dimension. Specifically the packet size could be $\{10, 25, 50, 75, 100, 250, 500, 750, 1000, 2500, 5000, 7500, 10000\}$ with a probability inversely proportional to the size itself, i.e. the smallest packet will benefit of an higher likelihood.

In the full buffer model we imposed that at each moment a packet is available to be transmitted with a size of 10 Mb, therefore the system is saturated from users' requests.

The last one considered, the video steaming model generates a video steaming session from real traces of the video *Big Buck Bunny*.

A. Training Phase

The learning algorithm was trained over 74 different simulations composed of 125 frames of 10 ms for a total of 9250 frames and simulated time 9.25 seconds. Traffic is generated using web browsing model and random ABS policy. Exploiting the off-policy property of the Q-learning algorithm, we ensure that most of all the possible states were visited several times and in such a way the values for the pair state-action could be well approximate.

B. Test Phase

The final test has been carried out in the same scenario, that is a simulation of 100 frames, where the users follow the same paths as depicted in figure 2; we compared the learned Q-policy with the benchmark ABS policies using all the possible traffic models.

V. RESULTS

As we can see in Fig. 3, the static policy shows the better performance in terms of average throughput of the user, the random has poor results with significant fluctuation and finally the Q-learning policy manifests a trend between the previous ones. Fig. 4 represents the worst scenario for the Q-policy because its throughput achieved is very low; the static policy proves to be stable over time instead the last one swings around the static policy. After a initial steep slope Q-learning and Static policies stabilize their throughput instead the random policy shows a minimum peak and then stabilizes itself too. However

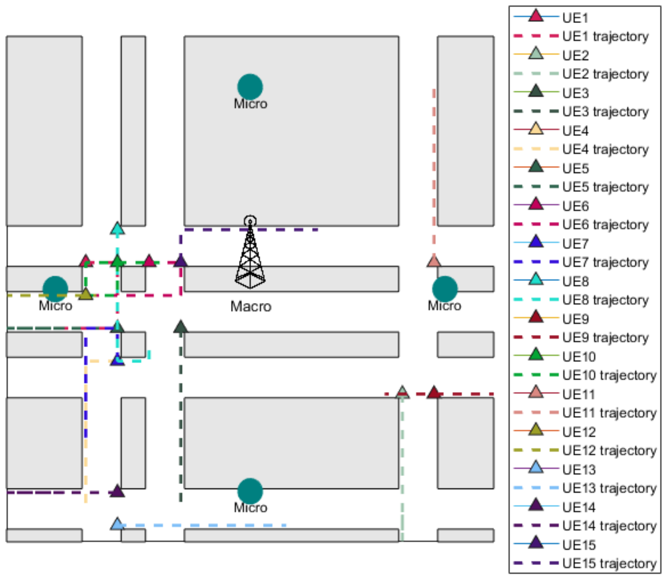

Figure 2: UEs' trajectories in the urban simulated environment during the final test; green dots represent the micros and the antenna the macro.

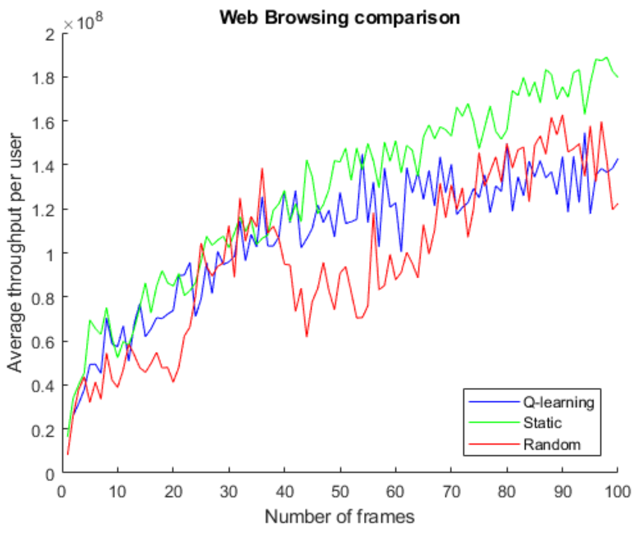

Figure 3: Evolution of the average throughput per user for each policy during the test simulation with Web Browsing traffic model.

Q-learning performs very poorly in this scenario. Analyzing the results it is clear that we did not achieve our purpose and this might be due to several factors. In particular we note an unexpected increasing throughput that could be due to simulator problems. Applying RL, we chose some parameters to summarize all the information needed to the agent to make decisions and it is possible that they are not enough representative of the environment's state not satisfying Markov Property. After that, because of the limited computational power and time that we had, we increased the users' speed in the simulation to make possible some handovers in our data and this can lead to unrealistic simulations. For the same reasons we had to limit

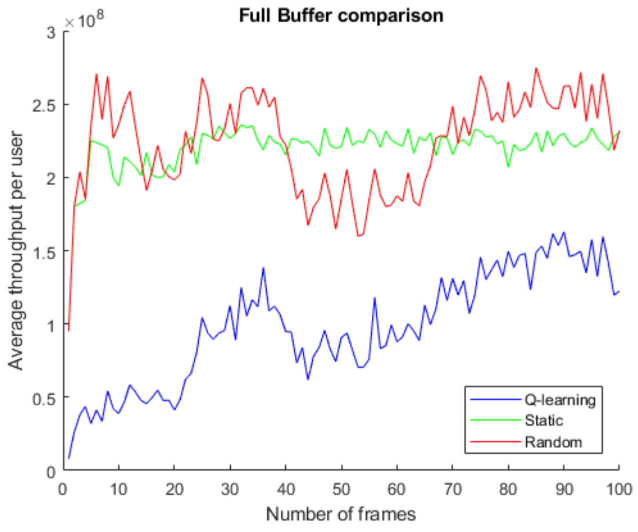

Figure 4: Evaluation of UEs' throughput for Full Buffer traffic model.

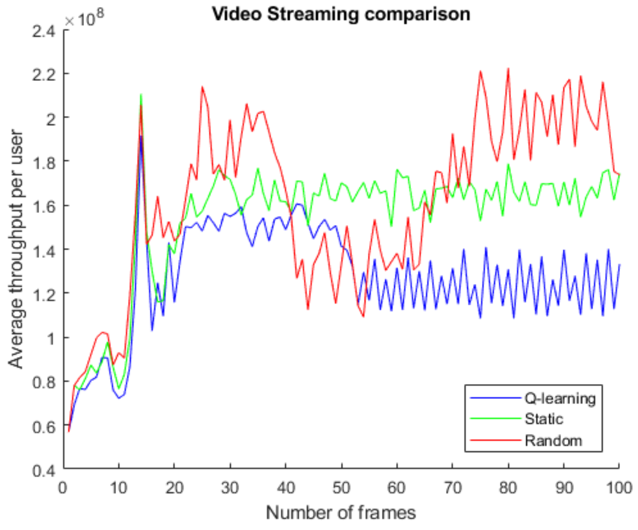

Figure 5: UEs' throughput profiles for Video Streaming model.

the total number of states limiting the number of parameters and also we had to quantize them in just some values. This can have also contributed to our bad results. Other problems could be related to learning phase: Q-learning convergence guarantee holds for an infinite number of visit for each state, a good approximation of Q-values could be obtained with a continuous update of states then 40 simulation of 100 frames each could be a misleading training set. During the training of our algorithm we choose the Web Traffic model since it shapes a more realistic scenario. Taking in account its packet generation randomness seams logically that it does not fit with the Full Buffer scenario as represented in Fig. 4. The last factor is the reward and chosen parameters, the former actually seems to us very reasonable but we may be wrong and not catch the learning problem, the latter instead could need a properly parameter selection.

VI. Conclusions

In this paper we examined the state of the art of the eICIC and then we tried to put in practice the Q-learning technique. Following the SON philosophy our contribution provides an attempt of exploit machine learning algorithms to auto-tune the network to UEs necessities. Our future work focus on tuning the Q-learning approach considering a different reward function and tuned parameters. In case of successful results with the target of improving performances, the use of Deep Q-Learning could be a good path to follow.

References

[1] Monster: a framework built around the matlab lte system toolbox. https://github.com/Sonohi/monster, 2017.
[2] Yvo de Jong Bultitude and Terhi Rautiainen. Ist-4-027756 winner ii d1. 1.2 v1. 2 winner ii channel models.
[3] Supratim Deb, Pantelis Monogioudis, Jerzy Miernik, and James P Seymour. Algorithms for enhanced inter-cell interference coordination (eicic) in lte hetnets. *IEEE/ACM transactions on networking*, 22(1):137–150, 2014.
[4] TS ETSI. 136 211. *Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 general aspects and principles (3GPP TS 36.420 version 14.0.1 Release 14)*, 2017.
[5] Cisco Visual Networking Index. Global mobile data traffic forecast update, 2016–2021 white paper, accessed on may 2, 2017.
[6] David Lopez-Perez, Ismail Guvenc, Guillaume De la Roche, Marios Kountouris, Tony QS Quek, and Jie Zhang. Enhanced intercell interference coordination challenges in heterogeneous networks. *IEEE Wireless Communications*, 18(3), 2011.
[7] Juan Ramiro and Khalid Hamied. *Self-organizing networks (SON): Self-planning, self-optimization and self-healing for GSM, UMTS and LTE.* John Wiley & Sons, 2011.
[8] Meryem Simsek, Mehdi Bennis, and Ismail Guvenc. Enhanced intercell interference coordination in hetnets: Single vs. multiflow approach. In *Globecom Workshops (GC Wkshps), 2013 IEEE*, pages 725–729. IEEE, 2013.
[9] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
[10] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.